(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
31 October 2002 (31.10.2002)

PCT

(10) International Publication Number
WO 02/087248 A2

(51) International Patent Classification⁷: H04N 7/24

(21) International Application Number: PCT/GB02/01796

(22) International Filing Date: 18 April 2002 (18.04.2002)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
01303562.1 19 April 2001 (19.04.2001) EP
60/305,846 18 July 2001 (18.07.2001) US

(71) Applicant *(for all designated States except US)*: **INDIGOVISION LIMITED** [GB/GB]; The Edinburgh Technopole, Bush Loan, Edinburgh EH26 0PJ (GB).

(72) **Inventors; and**
(75) **Inventors/Applicants** *(for US only)*: **SMART, Michael, Howard, William** [GB/GB]; 1F3, 11 Templepark Crescent, Edinburgh EH11 1JF (GB). **MACRAE, Donald, James** [GB/GB]; Flat 11, 23 Crosscauseway, Edinburgh EH8 9JW (GB). **DALZELL, Ryan** [GB/GB]; 8/11 Starbank Road, Edinburgh EH15 3BW (GB). **KEEPENCE, Barry** [GB/GB]; 8 Merchiston Place, Edinburgh EH10 4NP (GB). **BEATTIE, Robert, Scott** [GB/GB]; Flat 2, 29 West High Street, Lauder, Tweeddale TD2 6TF (GB).

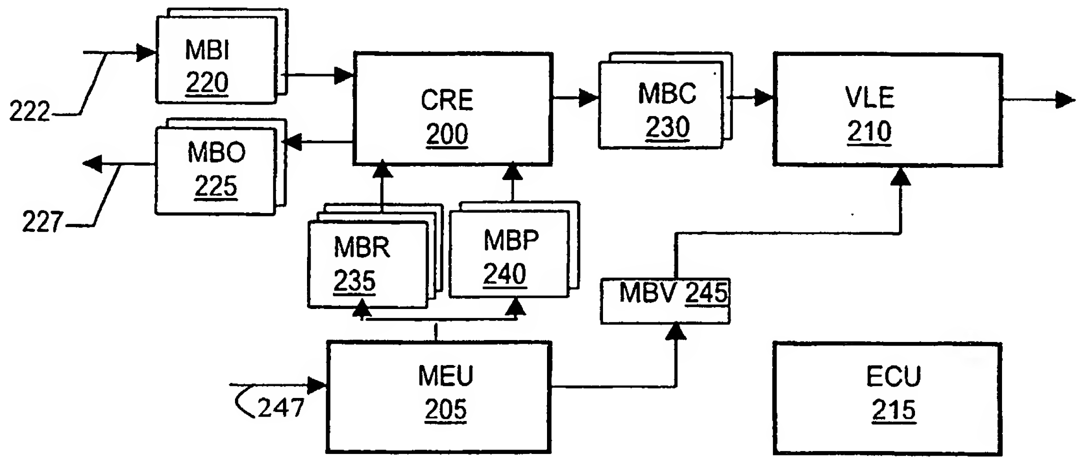
(74) **Agent: FITZPATRICKS**; 4 West Regent Street, Glasgow G2 1RS (GB).

(81) **Designated States** *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, OM, PH, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZM, ZW.

(84) **Designated States** *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent

[Continued on next page]

(54) Title: APPARATUS AND METHOD FOR PROCESSING VIDEO DATA



(57) **Abstract:** The present invention relates to processing hardware suitable for enabling an integrated chip (5) solution for decoding and/or encoding video data in accordance with a variety of video encoding formats, comprising a pipeline controller (55, 215) and a plurality of processing stages (15-30, 200-210) arranged in a pipeline for processing successive blocks of data corresponding to successive blocks of an image being processed, at least one of said processing stages (15-30, 200-210) being arranged to process a block of data with reference to data from a previously processed image, data arriving at each processing stage at regular intervals, where it is modified and passed on to the next stage, and so on. The pipeline controller (55, 215) may be arranged to specify one of a number of different encoding protocols supported by the pipeline processor, at least one of said processing stages (15-30, 200-210) being arranged to configure itself differently for compatibility with the specified protocol.

WO 02/087248 A2

(BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

— *without international search report and to be republished upon receipt of that report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

Apparatus And Method For Processing Video Data

The present invention relates to circuits and systems for processing video data. The invention in particular relates to processing hardware suitable for enabling an integrated chip solution to decoding and/or encoding video data in accordance with a variety of video encoding formats.

Digital video signals representing a sequence of images to form a motion picture are increasingly processed in the computer, entertainment and telecommunication arts. It is generally necessary to compress the "raw" video data and a variety of different encoding protocols have been defined according to particular requirements, and to reflect improvements in technology. Examples include Motion JPEG (M-JPEG), H.261, H.263 and the different MPEG protocols. The MPEG protocols are further evolving in different standards such as MPEG 1, 2 or 4. It is possible to implement the coding of a video bitstream into any such format using a variety of strategies, the decoding steps being fixed by the standard. In today's applications, decoding is often performed in real-time by the individual user, but the encoding may have been performed "off-line" and by a central information provider (TV station, DVD publisher etc). For other applications such as videoconferencing, it is essential that encoding and decoding are performed in real time. "Raw" video data in this context would typically mean video data in YUV pixel format. Y in this context represents luminance and U & V are chrominance components, often present at lower spatial resolution. Other formats such as RGB are equally known and data can be converted from one to the other. Apart from M-JPEG, the protocols rely on interframe coding, which exploits temporal consistency in the sequence of images, to achieve higher compression than can be achieved coding each picture in isolation (intraframe coding).

Coding and decoding may be carried out using software only, hardware, or some combination of software and hardware. An apparatus capable of coding and decoding is commonly termed a codec. Software-based solutions offer a high degree of freedom and adaptability regarding the coding format, whereas hardware based solutions offer advantages in speed at the expense of flexibility. However, due to computational demands of real-time video processing it is currently not feasible to implement a encoding and decoding solution in software only which can meet the quality requirements of users. Specialised processor hardware is available in chipsets which assist a host processor to achieve the necessary speed requirements, but suffer from several drawbacks.

Known hardware solutions suffer from a lack of flexibility, being dedicated to one of the above protocols. They are typically implemented to assist with the computationally intensive parts of the process, such as Discrete Cosine Transforms (DCT) and rely to a great extent on software control, typically in the form of a CPU controlling the coding/decoding process. In a real product, CPU cycles may have to be distributed among any of a number of different, non-video related tasks which will inevitably degrade codec performance. Product designers and programmers would prefer it if CPU time were free to provide the wider functionality and added value features required in a real product, and would rather not have CPU resources consumed by management of video coding.

Known encoding and decoding circuits generally require at least one frame store on chip, to support the interframe coding and decoding. The size of memory required for each stage in the process makes it difficult to integrate the entire coder/decoder on one chip, or to integrate the chip with a general-purpose processor core to form a "system on a chip" solution for handheld and other compact video products.

An example of a known pipeline architecture for video encoding is described in Ogura et al, "A 1.2-W single-Chip MPEG2 MP ML Video Encoder LSI Including Wide Search Range (H:+/-288,V:+/-96) Motion Estimation and 81-MOPS Controller", IEEE Journal of Solid-State Circuits, IEEE Inc., vol. 33, no. 11, November 1998. This is a single pipeline design with a multi-clock architecture where each processing entity is clocked differently. Interim data in this example is stored in external SDRAM. A second example of a pipeline architecture can be found in Chen et al, "Video Encoder Architecture For Real Time Encoding", IEEE Transactions on consumer electronics, IEEE Inc., vol 42, no. 3, 1 August 1996. This describes a three-stage pipeline for MPEG2 encoding wherein the video module inputs straight into the pipeline. In this example motion estimation is provided by a co-processor which is not part of the pipeline.

A further example of a pipeline architecture can be found in Fernandez et al, "A High Performance Architecture With Macroblock-Level-Pipeline for MPEG-2 Coding", Real-Time Imaging, Academic Press Limited, vol. 2, no. 6, 1 December 1996. Instead of a single bus to throughput the data, this design uses a "cross-bar network", imposing certain restrictions on interconnects due to the complexity of the network. This design also does not include the motion estimation functions, for which data flow requirements are even more onerous.

It is an object of the invention to enable the provision of a compact integrated circuit implementation of a video encoder and/or decoder, particularly but not exclusively one suitable for "system on a chip" integration with a general purpose processor core. Different aspects of the invention are defined, which aim variously at reducing on-chip memory, reducing the requirement for processor intervention, and/or providing a single codec configurable for different coding standards.

The invention provides a pipeline processor for processing digital data representing a sequence of images, each picture being divided for processing into a regular array of blocks of pixels, the processor being formed within an integrated circuit having an interface to external storage and comprising a pipeline controller and a plurality of processing stages arranged in a pipeline for processing successive blocks of data corresponding to successive blocks of an image being processed, at least one of said processing stages being arranged to process a block of data with reference to data from a previously processed image.

intervals, where it is modified and passed on to the next stage, and so on. The quantity of data within the pipeline at a given time is generally constant.

FIFO buffers may optionally be provided between the pipeline processor and the external storage interface, to decouple the timing of memory accesses from said systolic operation.

At least one intermediate pipeline stage may have access to said external storage interface, in addition to input and output processing stages, for the storage and retrieval of intermediate data. Said intermediate data may comprise said data from said previously processed image.

In a preferred embodiment, intermediate data written to said external storage during processing of a current image is not retrieved until processing of a subsequent image, any portion of said intermediate data required for processing of the current image being retained within the pipeline processor. The pipeline processor is preferably arranged to hold only a specific portion of the data representing the previously processed image at one time, corresponding to a specific part of the current image being processed at a given time.

The pipeline processor of the invention may be further distinguished by having any of the following sets of features, referred to here as aspects of the invention, whether alone or in combination.

In a first, more specific, aspect of the invention, the pipeline controller is arranged to operate in response to instructions from a program-controlled host processor, the pipeline controller controlling the fetching and processing of data for a picture on a block-by-block basis without block-by-block intervention from the host processor.

The pipeline controller, pipeline processing stages and internal storage may be arranged for systolic operation. FIFO buffers may be provided between the pipeline processor and the external memory interface, to decouple the timing of memory accesses from said systolic operation.

The pipeline controller may be responsive to an instruction specifying a source base location in said external storage, from which the location of all data for an image may be calculated. The instruction may further specify a destination base location for the output of processed data. The pipeline processor may have a DMA connection to the external storage.

The pipeline controller may be arranged to respond to an instruction from the host processor specifying one of a number of different encoding protocols supported by the pipeline processor, to configure and control the processing stages for compatibility with the specified protocol. For example, one protocol may permit motion vectors to be encoded with half-pixel precision, while another protocol permits only integer precision.

The pipeline controller may be arranged to respond to an instruction from the host processor specifying that a given image in the sequence is to be intraframe coded, that is without reference to previously processed images.

In a second aspect of the invention, the pipeline processor is arranged to store reconstructed image data in said external storage while processing a first image in the sequence, and to retrieve into internal storage successive parts of said reconstructed image data as said data from a previously processed image, while processing a subsequent image.

The pipeline processor may for example comprise stages for motion estimation and lossy encoding, together with a reconstruction pipeline for decoding and motion compensation, the reconstruction pipeline producing said reconstructed image data in parallel with the processing of the first image.

In such cases, the reconstructed image data may represent substantially an entire image, while the part retrieved at a given time represents only a restricted search area within the previously processed image, said part moving during processing, according to the block of the subsequent image currently being processed.

The reconstructed image data may be stored in a block format, as opposed to a whole-line raster format. This allows the block of data to be retrieved in a single DMA operation, rather than several separate runs of pixels.

In a third aspect of the invention, the pipeline processor comprises stages for applying motion estimation and predictive encoding to received image data, and further comprises a reconstruction pipeline for applying complementary decoding and motion compensation to the predictively encoded data to obtain reconstructed image data for the image being processed, the motion estimation stage being arranged to search for a best matching block of pixel data in a portion of reconstructed data generated and stored by said reconstruction pipeline during processing of said previously processed image, thereby to define a motion vector for use in said predictive encoding stage for a current block of pixels in the image being processed, the pipeline processor further comprising an on-chip store for holding, in a queue, the best matching block of pixel data found in the reconstructed data of the previously processed image as reference data for each block being processed, the motion compensation stage in the reconstruction pipeline being arranged to receive the held reference data from said queue at the same time as the decoded predictively encoded data for a given block, thereby to generate said reconstructed image data for the current frame without reference to externally stored data.

Use of the third aspect and second aspect of the invention in combination affords a particularly compact video compression encoder, having both limited on-chip storage requirement and limited bandwidth requirement for the interface to external storage.

In the case of a systolic pipeline processor, the capacity of the on-chip store for reference data may be fixed in accordance with the latency of the predictive encoding and decoding stages of the pipeline and reconstruction pipeline respectively.

The latency of the pipeline stages may depend upon a mode of operation selected from among plural possible modes, the length of said queue for reference data being adjusted accordingly.

In a fourth aspect of the aspect of the invention, the pipeline controller may be arranged to specify one of a number of different encoding protocols supported by the pipeline processor, at least one of said processing stages being arranged to configure itself differently for compatibility with the specified protocol.

A given pipeline processing stage may be arranged to change parameters of its operation according to the specified protocol. Alternatively or in addition, a given pipeline processing stage may be arranged to process data or to pass on data unmodified, depending on the specified protocol. Alternatively or in addition again, a given pipeline processing stage may be arranged to route data through physically different processing hardware depending on the specified protocol.

For example, one protocol (such as MPEG2) may permit motion vectors to be encoded with half-pixel precision, while another protocol (such as H.261) permits only integer precision. In this case, a half-pixel search processing stage may be disabled for certain protocols and enabled for others. In other variations, the range of motion vectors may need to be limited. Similarly for quantiser and variable length encoder stages, the range of permitted quantisation tables, coding sequences and so forth may be changed according to the protocol specified. All of these can be accommodated by appropriate configuration of the pipeline, without duplication of pipeline components.

The following optional features are appropriate to the above and other aspects of the invention generally. The processing stages may be arranged to perform an encoding process, in which the image data is received in a pixel-based format and processed into a quantised and variable-length coded block-based bitstream.

The processing stages may be arranged to perform a decoding process, in which the image data is received in the form of a quantised and variable-length coded block-based bitstream and processed into a pixel-based format.

Parallel pipeline processors may be provided for encoding and decoding two sequences of images in parallel, for example to permit a duplex video channel.

The pipeline processor may be integrated together with said host processor within said integrated circuit, but this is not essential. The host processor and pipeline processor may share access to the external storage.

The interface to the external storage may comprise a bus arrangement, said bus arrangement including separate interfaces to a plurality of said pipeline processing stages within the integrated circuit.

The blocks of pixels may comprise macroblocks including smaller blocks of luminance and chrominance data of different spatial resolutions, for example in compliance with MPEG specifications. Starting and ending processing stages within the pipeline may be arranged to operate on a macroblock basis, while intermediate stages operate on the individual blocks within the macroblocks.

BRIEF DESCRIPTION OF THE DRAWINGS

Embodiments of the invention will now be described, by way of example only, by reference to the accompanying drawings, in which:

Figure 1 shows schematically the architecture of a video pipeline processor adapted for implementation on an integrated chip;

Figure 2 is a block diagram of a video encoder pipeline processor architecture for processing a video sequence according to the present invention;

Figure 3 is a block diagram of a video encoder core within the video encoder pipeline of Figure 2;

Figure 4 is a block diagram of a motion estimator unit within the the encoder core of Figure 3;

Figure 5 is a block diagram of an encoder control unit within the encoder core of Figure 3;

Figure 6 is a diagram showing the organisation of processing elements of a pipeline for encoding and decoding a video sequence;

Figure 7 shows different types of generic pipeline entity and examples of possible modes of operation.

Figure 8 shows schematically the storage of video data as implemented by a video encoder pipeline processor architecture according to the present invention.

Appendix 1 includes timing charts explaining the operating sequence within the encoder of Figures 2 to 5.

DETAILED DESCRIPTION OF THE EMBODIMENTS

Figure 1 shows schematically the architecture of a video pipeline processor adapted for "system on a chip" (SOC) implementations of a video encoder and/or decoder. The form and function of the various components are specific to the different types of video encoder/decoder, and will not be described in relation to this Figure. Figure 1 does illustrate, however, the general relationship between the pipeline components, the controlling CPU and internal and external memory for storage of small and large amounts of data respectively. Video encoder and decoder pipelines, each of the form shown can be separately provided on the chip, as illustrated more explicitly in Figure 6. For the remainder of this description, it will be assumed that the encoder pipeline is represented in Figure 1, the decoder operating according to similar principles, but simplified. Other components such as audio codecs may also be provided in similar fashion, with or without software assistance from the controlling CPU.

In this example, a single integrated circuit 5 houses a central processing unit (CPU) 10 and real-time video encoder to form a "system on a chip". This CPU will typically be a RISC processor case such as ARM, SH2 (Hitachi) or the like, CPU 10 is regarded as the host processor, and typically has a number of other functions to manage in a real device. The video encoder comprises pipeline stages 15-30 (PS1 to PS4) with associated working storage formed by on-chip 35-45 (RAM1, RAM2, RAM3). IC 5 has two main interfaces to external components. CPU 10 is connected to a peripheral bus 50 such as ARM Peripheral Bus for the control of various peripheral devices (not shown) in a conventional manner. Via an internal extension of this bus, or directly, CPU 10 also issues instructions to pipeline controller 55 (PCTRL) which controls operation of the encoder pipeline. In the examples described further herein, this control is implemented on a frame-autonomous basis, while the individual pipeline stages operate on the picture data in one block (or macroblock) at a time. That is to say, the involvement of CPU 10 is limited to initiating the encoding of whole pictures or groups of pictures, while the dedicated controller 55 provides the sequencing necessary to steer all of the blocks/macroblocks of the picture(s) in turn through the pipeline process stages 15-30.

For encoding formats which employ temporal prediction (interframe coding), the pipeline stages require access to past or future pictures in the sequence, as well as data for the picture currently being encoded. Large quantities of picture data, even one complete frame, are not stored on IC 5, however, but are stored in external memory 60, while on-chip memory 35-45 handle only a small quantity of data. In the example illustrated, CPU 10 and the various pipeline stages are provided with individual DMA (direct memory access) interfaces 65-85 to the external memory, via a shared memory bus 87. Suitable bus technologies are known in the form of Double Data Rate (DDR) SDRAM bus and RAMBUS, for example. Each interface may have associated a FIFO (first-in, first-out) buffer within IC 5, to smooth out interruptions and bursts in the data streams. Depending on the specific applications and functionality the processor, some pipeline stages may not require external memory interfaces. This will be seen in the specific examples described below. It would also be possible for the different components to have independent external memory interfaces, where performance required exceeds that of a shared bus, but this is not necessary for the applications presently envisaged.

Outside IC 5 are a source 90 of raw video data, for example a local camera device, and a destination 95, for example a modem transmitting the encoded images to a remote terminal. These components are illustrated as having separate interfaces to the memory bus 87, which frees up CPU resources. Alternatively the source and/or destination could communicate with a different memory (not shown) or via the CPU's peripheral bus 50. The "raw" video data in this context would typically be video data in YUV format, each component Y, U, V being stored as a separate raster-based image plane. For the encoding processes mentioned, pixels are grouped into blocks of 8 x 8 pixels, and quartets of blocks are grouped into macroblocks of 16 x 16 pixels. As is well known in the art, Y values are typically stored for each pixel, while chrominance components U & V are stored with reduced resolution, with only 8 x 8 values each per macroblock.

Dashed arrows in Figure 1 illustrate the notional transfer paths of different types of data in the operation of the pipeline, all of which happen interleaved on a block-by-block basis via DMA interfaces 70-85. Transfer path T1 brings source image data from the source 90 to a source area within external RAM 60. Encoder pipeline stage 15 fetches this data (macro)block-by-(macro)block via path T2, to begin the encoding process. At a later stage in the encoding process, performed here by stage 20 (PS2), a substantial quantity of data is generated which will be required for the encoding of a subsequent picture. This data is transferred out to a working storage area within external RAM 60 via path T3, and then retrieved subsequently via path T4 when needed. Encoded bitstream data is transferred from the last pipeline stage 30 (PS4) to a destination area of external memory via path T5. This is then fetched from RAM 60 by the destination device 95, via path T6.

In parallel with the transfers of data to and from external RAM 60, various intermediate results are generated by the pipeline stages, which are required for the next or some subsequent stage within the same picture. Internal RAM buffers or registers are provided as appropriate. Block 45 (RAM3) represents a queuing path for data having a relatively long latency, but only within one picture coding period, which is kept on-chip for speed of access.

The data format and storage for processing thereof by the encoder processor of Figure 1 will be more fully explained with reference to subsequent Figures and attendant text.

Figure 2 is a block diagram of a video encoder pipeline processor representing a specific example of the type of system illustrated in Figure 1. The sequencing of the pipeline components on a block and macroblock basis is illustrated in the tables of Appendix 1, as will be explained more fully with reference to Figure 5. The encoder shown here is a generalised discrete cosine transform (DCT) based encoder with temporal predictive encoding, capable of implementing at least one encoder standard such as H.261, H.263, and MPEG codecs. The principal blocks shown are main encoder core CRE 200, the motion estimation unit MEU 205, variable length encoder unit VLE 210, and encoder control machine ECU 215. Also shown are macroblock input and output memory interfaces MBI, MBO 220, 225, macroblock-sized buffers MBC 230, MBP 235, MBR 240 and motion vector memory MBV 245. Associated with MEU 205 is a search data retrieval memory interface SCH 247, which can store up to nine macroblocks. The number of separate blocks shown indicates approximately the relative size of the memories.

The encoder shown in this example is designed to allow encoding of video in real-time according any one of a number of possible coding protocol. The actual function of the blocks shown, including the input and output of data, may differ according to the chosen coding protocol. The functionality of the principal blocks shown here is detailed later.

In this example the encoder receives video data, stored in a first instance as a macroblock by means of macroblock input memory interface MBI 220 Where the raw image is stored in a raster format, MBI performs the function of an address controller, producing the address in memory of each line of the macroblock for processing by the relevant stage of the pipeline. The source may alternatively be arranged to store image frames in external memory in macroblock format. The MBI 220 component of the pipeline produces the requests for retrieving a block or blocks of video data from over the memory bus and loading them into local memory for processing. Interface MBI 220 implements a transfer of type T1, as shown in Figure 1 using external memory interface 222 to retrieve stored data.

The main encoder core CRE 200 contains the specific processing entities equivalent to the generic pipeline stages 15-30 shown in Figure 1. The output of this block 200 is quantised data on a block-by-block basis, which is buffered by memory block MBC 230 to complete a macroblock, before processing by variable length encoder VLE 210. Encoder 210 encodes the quantised bitstream for output to memory according to the syntax specified by the coding. Core 200 also releases reconstructed macroblocks which are stored back into external memory via macroblock output interface MBO 225, to build up a complete reconstructed image for future use. As a result, the entire reconstructed image frame is then available to be used as a reference for predictively coding a subsequent image. Only those macroblocks actually being used are stored within the chip, however, and the complete image frames are stored only in external memory. Interface MBO 225 thus implements a transfer of type T3 in the generic pipeline of Figure 1 using external memory interface 227 to transfer data.. In the encoding of the subsequent image, MEU 205 uses another external memory interface 247 to retrieve parts of the stored data into a search buffer MSI 335, implementing a transfer of type T4. Assuming that the current macroblock should be interframe coded, motion estimator 205 receives data from a search area in a previously encoded image, to identify a best matching block of 16 x 16 pixels in the previously encoded image. The search area in this example represents an area of 3 x 3 macroblocks, centred on the current macroblock. For each new macroblock, only three macroblocks need to be retrieved.

The result of the search is a motion vector which, together with the reference image, predicts the current macroblock. The motion vectors are delayed in memory MBV 245 until they can be merged with the block data and encoded within the output bitstream by variable length encoder 210. At the end of the search, MEU 205 also saves the data of the best matching block of pixel data in buffer MBP 240, which can be used as reference data in the encoder core 200 for predictive encoding the current macroblock. This data can conveniently be stored in any internal format, rather than in the external raster format. The same data block is delayed for a longer latency period in buffer MBR 235, for use in a reconstruction process, to be described later with reference to Figure 3. Buffers 235 and 240 are thus examples of RAM 3 45 in the terminology of Figure 1.

The parameters for motion estimation may depend on the protocol selected. For example, it may only be necessary to perform searching at the integer level, as for H.261 type coding, or to perform searching at the half-pixel level, necessary for MPEG type coding. The designer is also free to determine the scope and quality of the motion estimation. For example, the search may be performed on Y data only (it being assumed that the U and V components are correlated in the same way), or the search area may be widened or restricted.

Figure 3 is a detailed block diagram of encoder core CRE 200 of the encoder pipeline shown in Figure 2. This shows the macroblock input and output interfaces MBI 220 and MBO 225 as before. The pipeline in fact comprises two pipelines for the processing of the video data, a "forward" pipeline 216, and an "inverse" pipeline 217. For encoding protocols which employ temporal prediction (e.g. interframe coding), the forward pipeline requires access to past or future pictures in the sequence, as well as data for the picture currently being encoded. The "forward" section 216 encodes macroblocks, while the "inverse" section 217 decodes the macroblocks, to reconstruct what will be recovered ultimately by a compatible decoder at a remote time or place. Because the encoding process is "lossy", what is reconstructed from the output bitstream will not correspond exactly with the images received from the source. The inverse pipeline 217 provides reconstructed macroblocks for use as the reference data for encoding subsequent images in the sequence.

Forward pipeline 216 input data from MBI 220 is passed to prediction unit PRE 260, then forward DCT unit 265 and forward quantisation unit 270. Similarly, the inverse pipeline 217 feeding MBO 225 includes reconstruction unit REC 275, inverse DCT unit IDC 280 and inverse quantisation unit 285. There are provided block buffers 290-310 (BB1-BB5) between the stages of the pipeline, analogous to the RAM buffers 35-45 of Figure 1. Other memory components include MBC 315 which buffers the output quantised coefficients from FQT 270 and buffers MBP 320 and MBR 325 which contain the predicted macroblock. Transposition memories BBF 330 and BBI 335 support the operation of DCT units FDC 265 and IDC 280 respectively, in a conventional manner.

In operation, uncompressed video data is processed taking a macroblock from input macroblock memory MBI 220 for processing by the prediction unit PRE 260. Prediction unit operates on each block within the macroblock and compares it with data from buffer MBP 320 which contains the predicted (best matching previous) macroblock found from the motion estimator unit MEU 205. Assuming that interframe coding is required for this macroblock, prediction unit PRE 260 passes only the prediction error data to the forward DCT component FDC 265, which in turn performs a spectral transform of the pixel data. The resultant DCT coefficients are zig-zag ordered and quantised and run length encoded in forward quantisation unit FQT 270. The resulting quantised data for all the blocks of the macroblock is accumulated in buffer MBC 230 before variable length encoding in unit VLE 210. Rate control, where required, is performed using bitstream statistics in a feedback loop to the quantiser, which is generic in its operation to allow for differences between video codec standards. Policies as to rate control are the subject of much design freedom, and the host CPU 10 may also be involved in this loop, without consuming much of its computational capacity.

As mentioned above, the feedback loop including inverse pipeline 217 allows motion estimation to be based upon what is reconstructed at a remote decoder, and compensates exactly the errors introduced by quantisation with time. Reconstruction unit REC 275 takes the reference pixel data provided by the MEU 205 and decoded error data from the IDC 280 and stores the reconstructed data back in memory, to be used as the next reference frame. This data is stored in macroblock format, for ease of retrieval and use within the encoder.

The number of layered blocks shown indicates broadly the relative sizes of the different on-chip memories. For example MBI 220 can be implemented as a dual ported memory capable of storing two macroblocks, while block buffer 295 is a dual ported memory capable of storing two blocks.

Figure 4 is a more detailed block diagram of motion estimator unit MEU 205. This shows the main elements including memory components MSI 335 and MSH 340 which are dual ported memories that contain the pixel data to be searched and MRI 345 and MRH 350 which are memories that contain the macroblock to be encoded next. The motion estimation unit is composed of two stages. The first stage (circuits with suffix I) carries out the integer pixel search and the second stage (suffix H) carries out the half-pixel search. To assist, the macroblock to be encoded is pre-shifted by a half pixel in x and/or y in memory MRH 350. MBV 355 is a dual ported memory that stores the motion vectors. MEI 360 is the integer estimation units and MEH 365 the half-pixel estimation unit.

The integer search stage employs an iterative log search strategy, where at each iteration 9 of the possible 15 x 15 possible (for a 15 x 15 search range) locations are tested starting with (x, y) = ((±8 and 0), (±8 and 0)) and halving the step size at each iteration. The absolute difference between pixel values in the macroblock to be encoded and the reference data being searched indicates the quality of the prediction for each possible motion vector.

The second search stage involves a half pixel (±0.5 and 0) search, using the result of the integer search as a starting point. This search stage outputs the motion vector, but also keeps the found best matching blocks of pixel data, which represent the prediction.

Pixel data to be searched is held in dual ported memory MSI 335 and the macroblock to be encoded held in memory MRI 345. The first stage integer pixel search is carried out by unit MEI 360. Data processed by this unit is stored in memory MSH and used by second stage unit MEH 365 for a half pixel search of the encoded block held in memory. Motion vector data about the macroblock is buffered at MBV 245 to be passed to the variable length encoder VLE 210. The prediction, which was found by the search among the reconstructed data of a previously encoded frame, is held in buffers MBP 320 and MBR 325 for use in encoder core CRE 200, within the forward and inverse sections of the pipeline 216, 217 respectively. The prediction is set to zero if it is decided to encode the macroblock as an intra block. This is usually done if the final absolute difference is large.

Note that with this design the two pipes, MEU 205 and CRE 200 are relatively well balanced, taking approximately the same number of clock ticks for each to complete processing. If the core (CRE) is speeded up, say by reducing the number of block ticks from 420 then this should be matched by an increase in motion estimator hardware. Similarly if latency through the core (CRE) is reduced then it will be necessary to provide a larger out store for the computed prediction.

The search area memory is a dual ported memory capable of storing 12 macro blocks. At any given time nine of these blocks are being used for motion estimation whilst the remaining three are being loaded from external memory. Additional loads are provided as necessary at the edges.

Figure 5 is a block diagram of an encoder control machine ECU 215. The principal entities of the controller unit are an encoder control unit ECC 400, memory control unit EMC 405 and macroblock formatter MBF 410.

The control unit ECC generates several main signals, which are illustrated also in Table 5 in Appendix 1. Firstly the memory switching signals SEL(0) to SEL(2) The signals determine whether the upper or lower part of a memory buffers address range is being written to, to implement double buffering. The signals are as follows:

- SEL(0) controls MBI, MBO, MBP, MBR, MRI, MRH, SAI & SAH, switches at the macroblock rate.
- SEL(1) controls BB1, BB2, BB3, BB4 & BB5, switches at the block rate.
- SEL(2) controls MBC, switches at the macroblock rate.

There are also four main memory load signals, LOD(0) to LOD(3), which determine which memory buffers require loading during any given macro block processing period. These signals are:

- LOD(0) initiates a search area memory load (MSI).
- LOD(1) initiates a reference memory (MRI, MRH) load.
- LOD(2) initiates an input memory (MBI) load.
- LOD(3) initiates a reconstruction memory (MBO) save.

Note that the design initially assumes that the output bitstream has its own interface to system memory. It could however be as easily combined with the core memory control described here. Finally the encoder control also generates the unit start signals, RUN(0) to RUN(3), which start the various units processing:

- RUN(0) starts the memory control unit (EMC)

- RUN(1) starts the motion estimation unit (MEU).
- RUN(2) starts the core units (CRE).
- RUN(3) starts the variable length encoder (VLE).
- STR the 1 bit start signal for the MEF
- DON the 1 bit done signal for the MEF

There are also some global signals, which will be employed in the overall control of the VLE and MEU units. These signals are set in or in response to an instruction from host processor 10, and govern the conduct of the processing for a complete image, as it is implemented by ECU 215. The principal global control signals are:

- FMT the 2 bit picture format, 1 = QCIF, 2=CIF, 3=4CIF.
- TYP the 2 bit encoder type, 0=H261, 1=H263, 2=MPEG4.
- INT the 1 bit INTRA flag.
- QNT the 5 bit quantizer value.
- CLK the 1 bit global clock.
- RST the 1 bit global reset.
- ENA the 1 bit clock enable.
- STA the one bit encoder start signal.
- RDY the one bit encoder ready signal.
- CBP the 6 bit coded block pattern (quantiser output).

The encoder control and memory management is in the charge of the two units ECC 400 and EMC 405. The encoder control unit is performs as a counter which counts the blocks and macroblocks. The block count is incremented every 420 clock ticks and the macroblock count every 6 blocks.

The actual control signals are then derived from the block and macroblock count signals. Note if it is necessary to stall the encoder because of delays, for example in filling or emptying the memory buffers, then the encoder can be paused by disabling transitions on the tick /block / macroblock counters.

The memory controller is based on a state machine which, based on the value of the LOD signals when the STA(0) is asserted high. The machine basically moves through its sequence of setting the number of the macroblock to be loaded on the input of the macroblock formatter MEF 410, establishing the correct write enable signals and then starting the MEF 410.

When the MEF 410 completes the EMC 405 state machine goes to its next state and repeats the process for the next memory to be loaded/stored. The exchange between the memory formatter MEF 410 and memory controller EMC 405 is done through two signals STR and DON. When all the memory transactions are complete the EMC unit asserts its RDY signal high which allows the encoder controller ECC 400 to increment its counters.

The predictor computation in the VLE 210 and search locations in the MEU 205 are based on internal counts of the number of (macro) blocks processed.

The reconstructed data is arranged in main system memory in a macroblock order. This arrangement is the most efficient in terms of paging faults and packing. The encoded data is output through it's own separate port and formatter (DMA) unit.

The CBP and INT flags generated by the forward quantiser FQT 270 and half pixel motion estimator MEH 365 are required by the variable length encoder VLE 210 and other units at different times and must therefore be buffered appropriately.

With appropriate sequencing, it is possible to replace double buffered memory components 290-310 with registers holding just one value each, reducing further the need for on chip RAM. This can also increase performance in terms of speed and pipeline latency, which is important in real-time applications. Decreasing the amount of on-chip RAM also increases the suitability of the pipeline for integration onto a single chip, as blocks of RAM introduce specific routing constraints in the chip layout process. Note that between some stages of the encoder pipeline the need for buffering may be removed altogether.

The encoder control machine ECU 215 shown here is equivalent to the pipeline controller PCTRL 55 of Figure 1. This component is dedicated to controlling the pipeline and encoding processes at the block or macroblock level allowing for decoupling of the encoding process from external processor intervention. At one level the encoder control machine ECU provides the necessary timing and data routing signal control ensuring that each component of the pipeline inputs and outputs video data in a timely fashion.

By use of an encoding control unit ECU to control the operation of the pipeline at a frame level it is possible to decouple the host CPU 10 from many of the supervisory roles that it would otherwise need to assume in the processing of the video data. The CPU need only oversee processing of the video data at a higher operational layer allowing the pipeline to operate in a semi-autonomous manner. The pipeline takes control of the encoding process at the frame level needing minimal intervention by external processing means. This frees the controlling CPU to devote processing cycles to other tasks.

The processor illustrated is fully pipelined and all units operate concurrently and with a common clock pulse (systolic operation). The memories are implemented in this example as dual ported (one read port one write port) devices. The pipeline is designed to be balanced such that each of the components consumes and produces data at equal rates, such that none of the components is starved of data. As the pipeline has no history of previous frames (that is, persistent data) it is possible to change coding protocols on a frame by frame basis. The timing for the pipeline is such that each stage of the pipeline processes data in a systolic fashion, balanced so that each stage is provided with data to process, processes this data and outputs data for processing for the next stage in a timely fashion. This reduces the latency of the system.

Appendix 1 shows example timing tables for the encoder core, motion estimator and encoder control signals.

The timing of the encoder is shown in Table 1 of Appendix 1. The corresponding contents of memory are shown in Table 2 of Appendix 1. Tables 1 and 2 show the progress of macroblocks through the different stages of the pipeline. The numbers in the table represent the macroblock and block that is being processed during a given time slice. The time slices are indexed on a MBC (macroblock count) / BLC (block count) pair. In MPEG protocols there are six blocks per macroblock. For the macroblock 3 the blocks are numbered (3,0) to (3,5) accordingly. Similarly, in Table 2 the numbers indicate the index of the macroblock that is being read from any given memory during the corresponding time slice.

For example, assuming a maximum data rate corresponding to CIF resolution (352 x 288) at 30 fps this means that 352 / 16 * 288 / 16 = 396 macroblocks or 396 * 6 = 2376 blocks must be processed per frame or every 1 / 30th of a second. If we assume a processor clock speed of 30MHz this allows approximately 1,000,000 ticks per frame of processing time or approximately 2520 ticks per macroblock(420 ticks per block).

To ensure the correct operation of the encoder at 30 MHz clock, no unit in the core must take longer than 420 ticks to process a block. Note that to process 4CIF at 30fps, without changing the hardware design, we must be able to clock the circuit at 120MHz or greater.

Tables 3 and 4 show the timing of the motion estimator relative to the encoder and the read memory contents during each tick. Note that the search are a memory is a dual ported memory capable of storing twelve macroblocks. At any given time nine of these blocks are being used for motion estimation whilst the final three are being loaded from external memory.

Depending on the arrangement of the actual motion estimator varying speeds of operation and complexity can be achieved.

Using nine search area data Sum of Absolute Differences (SAD) elements and assuming the data is 32 bit aligned requires approximately ((32 * 32) + (24 * 24) + (20

* 20) + (18 * 18) + (17 * 17) / 4) ≅ 700 (+ Overhead) clock ticks to compute the integer pixel search. Here we are assuming one DWORD of data is being fetched and processed each tick.

Using 1 SAD element on the other hand requires approximately (3 * 16 * 16 * 8 +16 * 16 * 9) / 4 ≅ 2200 (+ Overhead) cycles. Referring to the timing figures in the earlier section it is observed that this figure is within the maximum allowable tick count.

Tables 5 and 6 show the relative timing of the various control signals. It is assumed here that the individual units MEU, VLE and CTR are responsible for control of themselves. For example, the predictor computation in the VLE and search locations in the MEU are based on internal counts of the number of (macro) blocks processed.

Note that once the pipes, MEU and CRE are full, 1 input block, 3 search area blocks, 1 reference block, 1 reconstructed block and 1 chunk of encoded coefficients are loaded / saved to system memory every macroblock tick (2520 clock ticks). That is approximately 7 * 96 = 672, 32 bit read / writes or 7 * 48 = 336, 64 bit read / writes.

If the system memory and core are running at the same speed and each read / write takes one tick then there are used about 672 * 100 / 2520 = 26% (or 13% for 64 bit) of the available memory access units.

This access can be scheduled anywhere within the 2520 clock tick long macroblock processing time unit. However all loads and saves must be completed before the macroblock memory switch signals are activated or the encoder must be stalled.

Figures 6 and 7 are diagrams showing the organisation of processing elements of a pipeline processor for encoding and decoding video data in accordance with any one of a selection of possible coding protocols. This shows the implementation of the pipeline processor described above for use as a multi-codec pipeline which is capable of coding in any one of a number of possible video encoding protocols with minimal intervention by external processor means.

Figure 6 is a diagram showing the organisation of processing elements of a pipeline processor for processing a video bitstream. This presents a top-level overview of a generalised multi-codec encoder processor implementing the encoding apparatus discussed above and illustrated in Figures 1 through 5. The encoder pipeline can be represented as comprising three main elements: the encoder controller 445, "toolbox" entities 450 and encoder entities 455. The toolbox entities 450 and encoder entities are broadly equivalent to the first and second encoding stages of: 1) decomposition of a video bitstream into representative symbols; and 2) encoding of these symbols into binary strings. The encoder controller 445 here is broadly equivalent to the encoder controller PCTRL 55 and ECU 215 as detailed elsewhere.

The pipeline controller 445 may be arranged to specify one of a number of different encoding protocols supported by the pipeline processor, with the toolbox and encoding entities arranged to be configured for compatibility with the specified protocol.

The toolbox entities 450 comprise a number of processing elements necessary for the pre-processing of a video bitstream according to at least one encoding protocol. These components would typically comprise the coding core appropriate to the specified protocols and peripheral elements such as memory components and motion estimation units. For example, typical entities for MPEG compliant encoder machine would include the various components as set out in Figure 3, including memory components, producing run length encoded video bitstream data. These entities would be configured to encode a video bitstream compliant with the specified protocol, the encoder controller 445 configuring the individual elements needed to operate accordingly. Individual processor entity types are detailed later.

The encoder controller 445 specifies connections and data routing so that the appropriate entities comprising the elements of a single codec protocol are connected so as to receive, process and output data according to a specified protocol in a timely fashion. By configuring the various entities required for a specific encoding protocol and connections therebetween in a balanced and optimised fashion it is possible to construct a pipeline processor such as that illustrated in figures 1 to 5, capable of encoding an input video bitstream in a systolic fashion and optimised for implementation as a single "system on a chip".

The codec set of entities 455 comprises the processing elements needed for the variable length coding of the video data as pre-processed the toolbox entities. This is essentially equivalent to the variable length encoding component VLE 210 of figure 2 and performs the necessary encoding of the run-length encoded data output from the toolbox entities.
Also shown in figure 6 is the decoding of a video stream which has been encoded according to any one of a variety of protocols. This similarly comprises a decoder controller and collection of toolbox and decoder entities 480, 485. An input encoded bitstream 490 is decoded at 485, passed 495 to the toolbox entities 480 and output 500 as video data. Both the encoder and decoder shown here may be integrated in a single integrated chip solution.

Figure 7 is shows different types of pipeline entity and examples of possible modes of operation. Three types of entity are shown here: type-A 505, type-B 510 and type-C 540. Each type of entity has an input 515, and output 520 and control signal 525.

In a type-A entity 505, the entity operation is codec protocol independent. This means that all data received by a type-A entity 505 will be processed in exactly the same manner, regardless of with which protocol the data will be encoded.

In a type-B entity 510, the entity operation is codec protocol dependent. The data received by a type-B entity 510 will be processed according to protocol specific parameters instructed by control signal 525 from the encoder controller 445. Examples of such entities would be those performing quantisation functions.

The third type of entity, type-C 515, allows for routing behaviour to be controlled. Sub-entities may be encapsulated within the entity 515. Here two such sub-entities 530, 535 are shown but the number may vary. A control signal designates the entity behaviour to either route data input into one of the two entities 530, 535 for processing, or to bypass the sub-entities without any processing on the data being carried out.

The combination of behaviour and sub-behaviour of the entities as controlled by pipeline controller 445/475 allows for the routing of data through the appropriate processing stages to encode/decode a video bitstream according to a specified protocol.

Figure 8 illustrates schematically how the video data is organised in memory. Frames of video data are held in external memory 600, broadly equivalent here to the external memory EXTRAM 60 detailed earlier. The format in which the video data is stored is an important consideration regarding the speed at which an encoder can access and process that data.

With reference to the preceding Figures, the raw video is input into the encoder in YUV format. The YUV format stores video data as a luminance and two colour difference signals. This type of video data is stored in memory in a planar format, that is all the bits encoding the luminance signal Y stored in consecutive memory locations, all the bits encoding the colour difference signal U stored in consecutive memory locations, and all the bits encoding the colour difference signal V stored in consecutive memory locations. This manner of storage for a memory signal brings with it certain disadvantages when constant access to and manipulation of the data is required. An encoder must apply processing effort in locating data corresponding to blocks of pixels for encoding in macroblock format.

During coding the video bitstream is converted into a macroblock format. In general terms, each frame of the video bitstream is divided into blocks of pixels (which may comprise macroblocks including smaller blocks) and the bits representing the YUV components stored in memory, organised according to the macroblocks of the frame they represent. Macroblocks are numbered and stored consecutively, starting with macroblock 1. This format of video data storage allows the encoder to rapidly access data for processing as the data required for processing is stored contiguously, not distributed in different memory locations as in the case of YUV planar data.

In the encoder processor described previously with reference to Figures 1 to 8 and appendices, the YUV input data is converted into macroblock format at an early stage of the encoder processing cycle. The frames required for processing are stored in external memory and can be accessed as needed by the encoder processor elements. . Storing data off-chip in a readily accessible macroblock format. reduces the latency of the elements processing the data as no extra processing cycles are needed to locate the different segments of data, representing the Y, U and V components of the frame. An example of suitable processor for implementing this type of devolved data access is illustrated and described in Figure 1.

Those skilled in the art will appreciate that the embodiments described above are presented by way of example only, and that many further modifications and variations are possible within the spirit and scope of the invention.

APPENDIX 1 – TIMING TABLES

T	3, 0	3, 1	3, 2	3, 3	3, 4	3, 5	4, 0	4, 1	4, 2	4, 3	4, 4	4, 5	5, 0	5, 1	5, 2	5, 3	5, 4	5, 5	6, 0	6, 1	6, 2	6, 3	6, 4	6, 5	7, 0	7, 1	7, 2	7, 3	7, 4	7, 5
PRE	0, 0	0, 1	0, 2	0, 3	0, 4	0, 5	1, 0	1, 1	1, 2	1, 3	1, 4	1, 5	2, 0	2, 1	2, 2	2, 3	2, 4	2, 5	3, 0	3, 1	3, 2	3, 3	3, 4	3, 5	4, 0	4, 1	4, 2	4, 3	4, 4	4, 5
FDC		0, 0	0, 1	0, 2	0, 3	0, 4	0, 5	1, 0	1, 1	1, 2	1, 3	1, 4	1, 5	2, 0	2, 1	2, 2	2, 3	2, 4	2, 5	3, 0	3, 1	3, 2	3, 3	3, 4	3, 5	4, 0	4, 1	4, 2	4, 3	4, 4
FQT			0, 0	0, 1	0, 2	0, 3	0, 4	0, 5	1, 0	1, 1	1, 2	1, 3	1, 4	1, 5	2, 0	2, 1	2, 2	2, 3	2, 4	2, 5	3, 0	3, 1	3, 2	3, 3	3, 4	3, 5	4, 0	4, 1	4, 2	4, 3
IQT				0, 0	0, 1	0, 2	0, 3	0, 4	0, 5	1, 0	1, 1	1, 2	1, 3	1, 4	1, 5	2, 0	2, 1	2, 2	2, 3	2, 4	2, 5	3, 0	3, 1	3, 2	3, 3	3, 4	3, 5	4, 0	4, 1	4, 2
IDC					0, 0	0, 1	0, 2	0, 3	0, 4	0, 5	1, 0	1, 1	1, 2	1, 3	1, 4	1, 5	2, 0	2, 1	2, 2	2, 3	2, 4	2, 5	3, 0	3, 1	3, 2	3, 3	3, 4	3, 5	4, 0	4, 1
REC							0, 0	0, 1	0, 2	0, 3	0, 4	0, 5	1, 0	1, 1	1, 2	1, 3	1, 4	1, 5	2, 0	2, 1	2, 2	2, 3	2, 4	2, 5	3, 0	3, 1	3, 2	3, 3	3, 4	3, 5

TABLE 1: Core timing. The number(s) in the table represent the macroblock and block that is being processed during a given time slice. The time slices are indexed on a mbc (macroblock count) / blc (block count) pair.

T	3, 0	3, 1	3, 2	3, 3	3, 4	3, 5	4, 0	4, 1	4, 2	4, 3	4, 4	4, 5	5, 0	5, 1	5, 2	5, 3	5, 4	5, 5	6, 0	6, 1	6, 2	6, 3	6, 4	6, 5	7, 0	7, 1	7, 2	7, 3	7, 4	7, 5
MBI	0	0	0	0	0	0	1	1	1	1	1	1	2	2	2	2	2	2	3	3	3	3	3	3	4	4	4	4	4	4
MBO													0	0	0	0	0	0	1	1	1	1	1	1	2	2	2	2	2	2
MBC									0	0	0	0	0	0	1	1	1	1	1	1	2	2	2	2	2	2	3	3	3	3
MBP	0	0	0	0	0	0	1	1	1	1	1	1	2	2	2	2	2	2	3	3	3	3	3	3	4	4	4	4	4	4
MBR							0	0	0	0	0	0	1	1	1	1	1	1	2	2	2	2	2	2	3	3	3	3	3	3

TABLE 2: Memory contents. The numbers in the table above indicate the index of the macroblock that is being read from any given memory during the corresponding time slice.

T	3, 0	3, 1	3, 2	3, 3	3, 4	3, 5	4, 0	4, 1	4, 2	4, 3	4, 4	4, 5	5, 0	5, 1	5, 2	5, 3	5, 4	5, 5	6, 0	6, 1	6, 2	6, 3	6, 4	6, 5	7, 0	7, 1	7, 2	7, 3	7, 4	7, 5
PRE	0, 0	0, 1	0, 2	0, 3	0, 4	0, 5	1, 0	1, 1	1, 2	1, 3	1, 4	1, 5	2, 0	2, 1	2, 2	2, 3	2, 4	2, 5	3, 0	3, 1	3, 2	3, 3	3, 4	3, 5	4, 0	4, 1	4, 2	4, 3	4, 4	4, 5
FDC	*	0, 0	0, 1	0, 2	0, 3	0, 4	0, 5	1, 0	1, 1	1, 2	1, 3	1, 4	1, 5	2, 0	2, 1	2, 2	2, 3	2, 4	2, 5	3, 0	3, 1	3, 2	3, 3	3, 4	3, 5	4, 0	4, 1	4, 2	4, 3	4, 4
FQT	*	*	0, 0	0, 1	0, 2	0, 3	0, 4	0, 5	1, 0	1, 1	1, 2	1, 3	1, 4	1, 5	2, 0	2, 1	2, 2	2, 3	2, 4	2, 5	3, 0	3, 1	3, 2	3, 3	3, 4	3, 5	4, 0	4, 1	4, 2	4, 3
IQT	*	*	*	0, 0	0, 1	0, 2	0, 3	0, 4	0, 5	1, 0	1, 1	1, 2	1, 3	1, 4	1, 5	2, 0	2, 1	2, 2	2, 3	2, 4	2, 5	3, 0	3, 1	3, 2	3, 3	3, 4	3, 5	4, 0	4, 1	4, 2
IDC	*	*	*	*	0, 0	0, 1	0, 2	0, 3	0, 4	0, 5	1, 0	1, 1	1, 2	1, 3	1, 4	1, 5	2, 0	2, 1	2, 2	2, 3	2, 4	2, 5	3, 0	3, 1	3, 2	3, 3	3, 4	3, 5	4, 0	4, 1
REC	*	*	*	*	*	*	0, 0	0, 1	0, 2	0, 3	0, 4	0, 5	1, 0	1, 1	1, 2	1, 3	1, 4	1, 5	2, 0	2, 1	2, 2	2, 3	2, 4	2, 5	3, 0	3, 1	3, 2	3, 3	3, 4	3, 5
MEI	2	2	2	2	2	2	3	3	3	3	3	3	4	4	4	4	4	4	5	5	5	5	5	5	6	6	6	6	6	6
MEH	1	1	1	1	1	1	2	2	2	2	2	2	3	3	3	3	3	3	4	4	4	4	4	4	5	5	5	5	5	5

TABLE 3: Core and motion estimator timing. The number(s) in the table represent the macroblock and block that is being processed during a given time slice.

T	3, 0	3, 1	3, 2	3, 3	3, 4	3, 5	4, 0	4, 1	4, 2	4, 3	4, 4	4, 5	5, 0	5, 1	5, 2	5, 3	5, 4	5, 5	6, 0	6, 1	6, 2	6, 3	6, 4	6, 5	7, 0	7, 1	7, 2	7, 3	7, 4	7, 5
MBI	0	0	0	0	0	0	1	1	1	1	1	1	2	2	2	2	2	2	3	3	3	3	3	3	4	4	4	4	4	4
MBO	*	*	*	*	*	*	*	*	*	*	*	*	0	0	0	0	0	0	1	1	1	1	1	1	2	2	2	2	2	2
MBC	*	*	*	*	*	*	*	*	0	0	0	0	0	0	1	1	1	1	1	1	2	2	2	2	2	2	3	3	3	3
MBP	0	0	0	0	0	0	1	1	1	1	1	1	2	2	2	2	2	2	3	3	3	3	3	3	4	4	4	4	4	4
MBR	*	*	*	*	*	*	0	0	0	0	0	0	1	1	1	1	1	1	2	2	2	2	2	2	3	3	3	3	3	3
SAI	2	2	2	2	2	2	3	3	3	3	3	3	4	4	4	4	4	4	5	5	5	5	5	5	6	6	6	6	6	6
SAH	1	1	1	1	1	1	2	2	2	2	2	2	3	3	3	3	3	3	4	4	4	4	4	4	5	5	5	5	5	5

TABLE 4: Macroblock read memory contents. The numbers in the table above indicate the index of the macroblock that is being read from any given memory during the corresponding time slice.

T	0,0	0,1	0,2	0,3	0,4	0,5	1,0	1,1	1,2	1,3	1,4	1,5	2,0	2,1	2,2	2,3	2,4	2,5	3,0	3,1	3,2	3,	3,4	3,5	4,0	4,1	4,2	4,3	4,4	4,5	5,0
LOD																															
0	1						1						1						1						1						1
1													1						1						1						1
2																															1
3																															
RUN																															
0							1						1						1						1						1
1																			1	1	1	1	1	1	1	1	1	1	1	1	1
2																											1				
SEL																															
0	1						1						1						1						1						1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2			1						1						1						1						1				

TABLE 5: timing related to ECC. Control signals are held high for one clock period at the beginning of each time slice.

T	0,0	1,0	2,0	3,0	4,0	5,0	6,0
	Load SAI 0	Load SAI 1	Load SAI 2	Load SAI 3	Load SAI 4	Load SAI 5	Load SAI 6
		Compute MEI 0	Compute MEI 1	Compute MEI 2	Compute MEI 3	Compute MEI 4	Compute MEI 5
			Compute MEH 0	Compute MEH 1	Compute MEH 2	Compute MEH 3	Compute MEH 4
			Load MBI 0	Load MBI 1	Load MBI 2	Load MBI 3	Load MBI 4
				Compute CRE 0	Compute CRE 1	Compute CRE 2	Compute CRE 3
					Compute VLE 0	Compute VLE 1	Compute VLE 2
						Save MBO 0	Save MBO 1

TABLE 6: The table above shows the sequence of events after the encoder is given the start signal.

CLAIMS

1. A pipeline processor for processing digital data representing a sequence of images, each picture being divided for processing into a regular array of blocks of pixels, the processor being formed within an integrated circuit (5) having an interface to external storage (60) and comprising a pipeline controller (55) and a plurality of processing stages (15, 20, 25, 30) arranged in a pipeline for processing successive blocks of data corresponding to successive blocks of an image being processed, at least one of said processing stages being arranged to process a block of data with reference to data from a previously processed image.

2. A pipeline processor as claimed in claim 1 wherein the pipeline controller, pipeline stages and internal storage are arranged for systolic operation wherein data arrives at each processing stage at regular intervals, where it is modified and passed on to the next stage, the quantity of data within the pipeline at a given time being generally constant.

3. A pipeline processor as claimed in claims 1 or 2 wherein FIFO buffers are provided between the pipeline processor and the external storage interface, to decouple the timing of memory accesses from said systolic operation.

4. A pipeline processor as claimed in any preceding claim wherein at least one intermediate pipeline stage has access to said external storage interface, in addition to input and output processing stages, for the storage and retrieval of intermediate data where said intermediate data are comprise said data from said previously processed image.

5. A pipeline processor as claimed in any preceding claim wherein intermediate data written to said external storage during processing of a current image is not retrieved until processing of a subsequent image, any portion of said intermediate data required for processing of the current image being retained within the pipeline processor, the pipeline processor preferably arranged to hold only a specific portion of

the data representing the previously processed image at one time, corresponding to a specific part of the current image being processed at a given time.

6. A pipeline processor as claimed in any preceding claim wherein the pipeline controller (55) is arranged to operate in response to instructions from a program-controlled host processor (10), the pipeline controller (55) controlling the fetching and processing of data for a picture on a block-by-block basis without block-by-block intervention from the host processor.

7. A pipeline processor as claimed in claim 6 wherein the pipeline controller, pipeline processing stages and internal storage are arranged for systolic operation wherein FIFO buffers are provided between the pipeline processor and the external memory interface, to decouple the timing of memory accesses from said systolic operation.

8. A pipeline processor as claimed in claim 6 or 7 wherein the pipeline controller is responsive to an instruction specifying a source base location in said external storage, from which the location of all data for an image is calculated and where the instruction may further specify a destination base location for the output of processed data, the pipeline processor having a DMA connection to the external storage.

9. A pipeline processor as claimed in claim 6, 7 or 8 wherein the pipeline controller is arranged to respond to an instruction from the host processor specifying one of a number of different encoding protocols supported by the pipeline processor, to configure and control the processing stages for compatibility with the specified protocol.

10. A pipeline processor as claimed in claim 6, 7, 8 or 9 wherein one protocol permits motion vectors to be encoded with half-pixel precision, while another protocol permits only integer precision.

11. A pipeline processor as claimed in claim 6, 7, 8, 9, 10 wherein the pipeline controller is arranged to respond to an instruction from the host processor specifying that a given image in the sequence is to be intraframe coded, that is without reference to previously processed images.

12. A pipeline processor as claimed in any preceding wherein claim the pipeline processor is arranged to store reconstructed image data in said external storage (60) while processing a first image in the sequence, and to retrieve into internal storage (35, 40, 45) successive parts of said reconstructed image data as said data from a previously processed image, while processing a subsequent image.

13. A pipeline processor as claimed in claim 12 wherein the pipeline processor comprises stages for motion estimation and lossy encoding, together with a reconstruction pipeline for decoding and motion compensation, the reconstruction pipeline producing said reconstructed image data in parallel with the processing of the first image.

14. A pipeline processor as claimed in claim 12 or 13 wherein the reconstructed image data represents substantially an entire image, while the part retrieved at a given time represents only a restricted search area within the previously processed image, said part moving during processing, according to the block of the subsequent image currently being processed.

15. A pipeline processor as claimed in claim 12, 13 or 14 wherein the reconstructed image data is stored in a block format, as opposed to a whole-line raster format allowing the block of data to be retrieved in from a contiguous block of memory locations, rather than several separate runs of pixel locations.

16. A pipeline processor as claimed in any preceding claim wherein the pipeline processor comprises stages (205, 360, 365) for applying motion estimation and predictive encoding to received image data, and further comprises a reconstruction pipeline (217) for applying complementary decoding and motion compensation to the

predictively encoded data to obtain reconstructed image data for the image being processed, the motion estimation stage (205) being arranged to search for a best matching block of pixel data in a portion of reconstructed data generated and stored by said reconstruction pipeline during processing of said previously processed image, thereby to define a motion vector for use in said predictive encoding stage for a current block of pixels in the image being processed, the pipeline processor further comprising an on-chip store for holding, in a queue, the best matching block of pixel data found in the reconstructed data of the previously processed image as reference data for each block being processed, the motion compensation stage (205) in the reconstruction pipeline (217) being arranged to receive the held reference data from said queue at the same time as the decoded predictively encoded data for a given block, thereby to generate said reconstructed image data for the current frame without reference to externally stored data.

17. A pipeline processor as claimed in claim 16 wherein providing a compact video compression encoder, having both limited on-chip storage requirement and limited bandwidth requirement for the interface to external storage.

18. A pipeline processor as claimed in claim 16 or 17 wherein for the case of a systolic pipeline processor, the capacity of the on-chip store for reference data is fixed in accordance with the latency of the predictive encoding and decoding stages of the pipeline and reconstruction pipeline respectively.

19. A pipeline processor as claimed in claim 16, 17 or 18 wherein the latency of the pipeline stages depends upon a mode of operation selected from among plural possible modes, the length of said queue for reference data being adjusted accordingly.

20. A pipeline processor as claimed in any preceding claim wherein the pipeline controller (55) is arranged to specify one of a number of different encoding protocols supported by the pipeline processor, at least one of said processing stages (15-30) being arranged to configure itself differently for compatibility with the specified protocol.

21. A pipeline processor as claimed in claim 20 wherein a given pipeline processing stage is arranged to change parameters of its operation according to the specified protocol such that a given pipeline processing stage is arranged to process data or to pass on data unmodified, depending on the specified protocol, or is arranged to route data through physically different processing hardware, depending on the specified protocol.

22. A pipeline processor as claimed in any preceding claim wherein the processing stages are arranged to perform an encoding process, in which the image data is received in a pixel-based format and processed into a quantised and variable-length coded block-based bitstream.

23. A pipeline processor as claimed in any of claims 1 to 21 wherein the processing stages are arranged to perform a decoding process, in which the image data is received in the form of a quantised and variable-length coded block-based bitstream and processed into a pixel-based format.

24. A pipeline processor as claimed in any preceding claim wherein parallel pipeline processors are provided for encoding and decoding two sequences of images in parallel, for example to permit a duplex video channel.

25. A pipeline processor as claimed in any preceding claim wherein the pipeline processor is integrated together with said host processor within said integrated circuit.

26. A pipeline processor as claimed in any preceding claim wherein the host processor and pipeline processor share access to the external storage.

27. A pipeline processor as claimed in any preceding claim wherein the interface to the external storage comprises a bus arrangement, said bus arrangement including separate interfaces to a plurality of said pipeline processing stages within the integrated circuit.

28. A pipeline processor as claimed in any preceding claim wherein the blocks of pixels comprise macroblocks including smaller blocks of luminance and chrominance data of different spatial resolutions.

29. A pipeline processor as claimed in any preceding claim wherein starting and ending processing stages within the pipeline are arranged to operate on a macroblock basis, while intermediate stages operate on the individual blocks within the macroblocks.

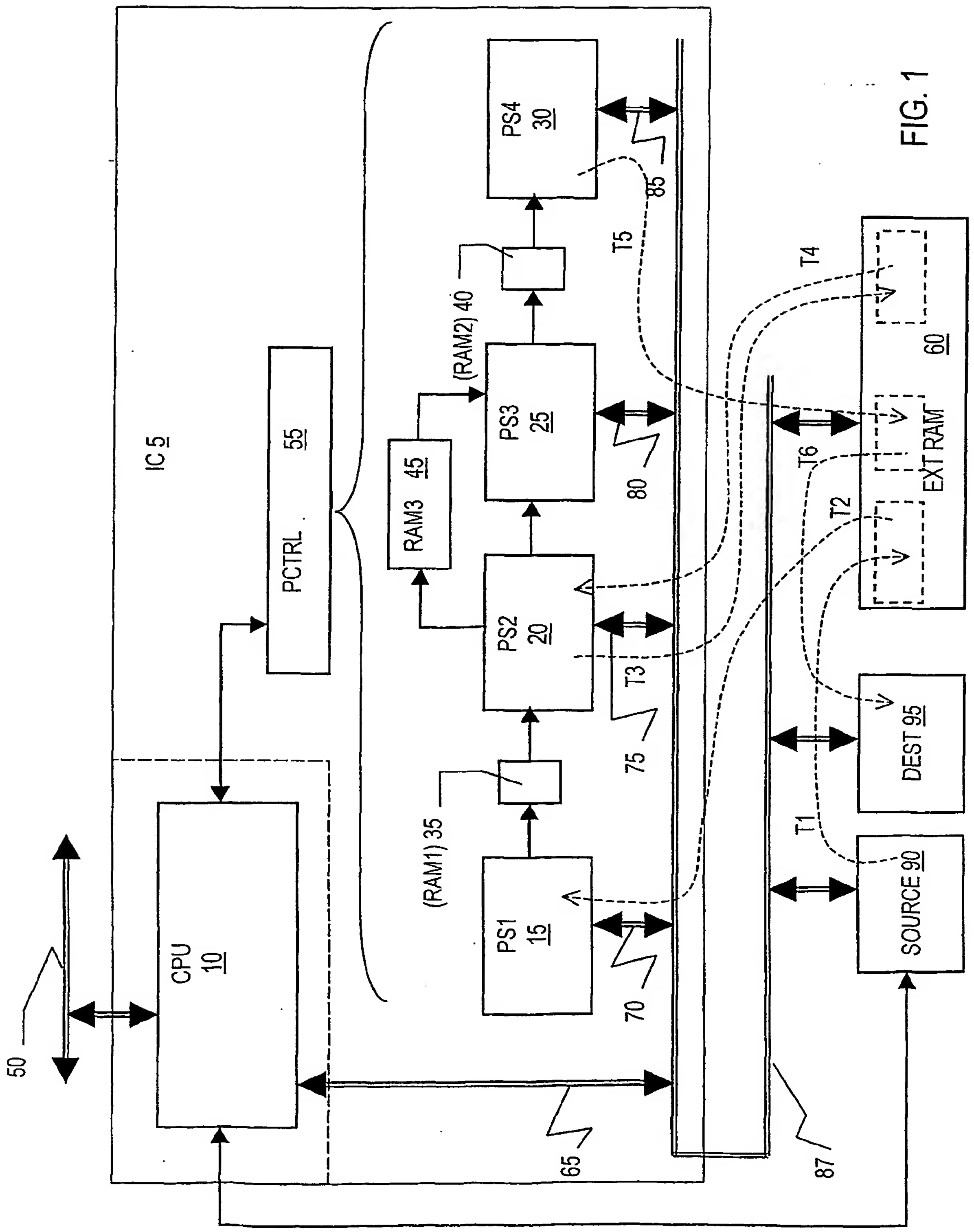

IC 5
PCTRL 55
CPU 10
PS1 15
PS2 20
PS3 25
PS4 30
(RAM1) 35
(RAM2) 40
RAM3 45
50
65
70
75
80
85
87
T1
T2
T3
T4
T5
T6
EXT RAM 60
DEST 95
SOURCE 90

FIG. 1

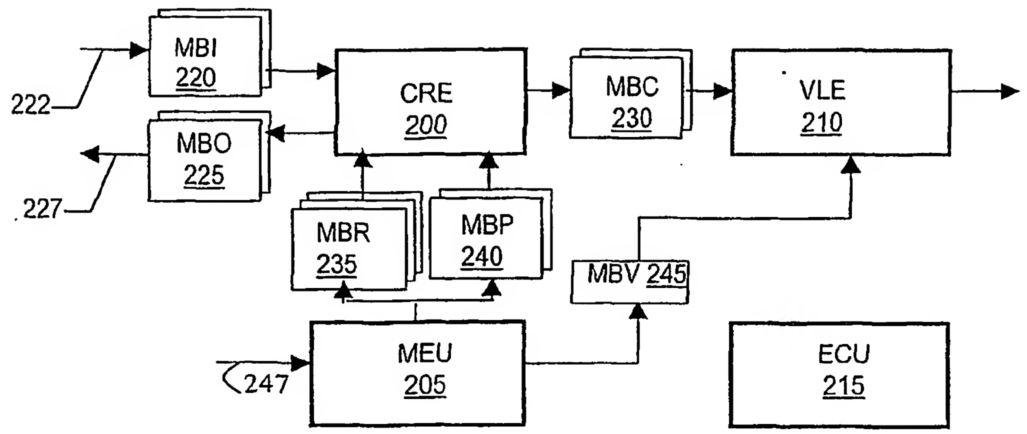

MBI
220
CRE
200
MBC
230
VLE
210
222
MBO
225
227
MBR
235
MBP
240
MBV 245
MEU
205
247
ECU
215

FIG. 2

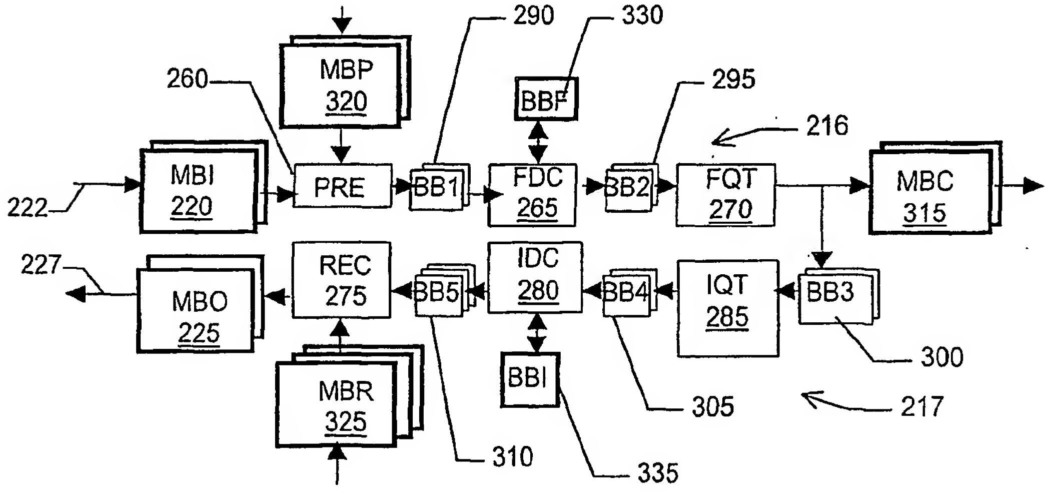

290
330
MBP
320
260
BBF
295
216
MBI
220
PRE
BB1
FDC
265
BB2
FQT
270
MBC
315
222
227
REC
275
IDC
280
MBO
225
BB5
BB4
IQT
285
BB3
300
BBI
MBR
325
305
217
310
335

FIG. 3

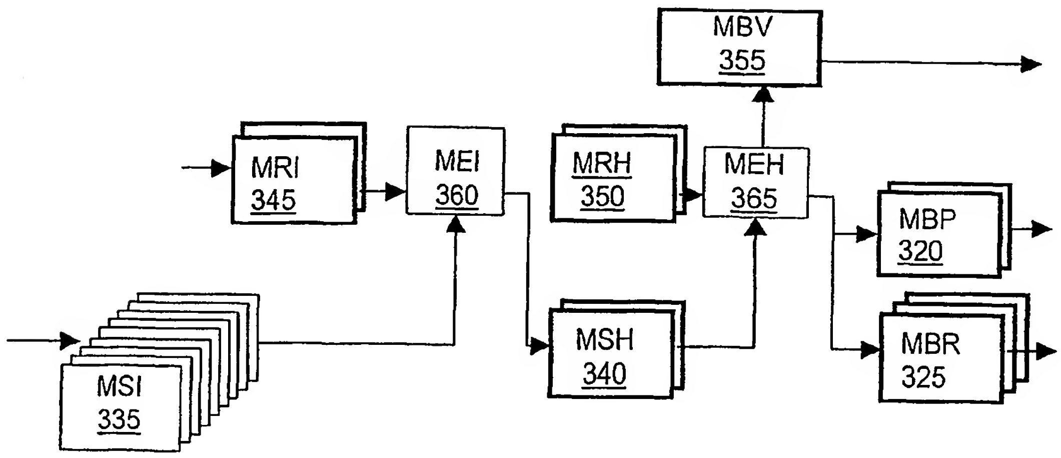

MBV
355
MRI
345
MEI
360
MRH
350
MEH
365
MBP
320
MSI
335
MSH
340
MBR
325

FIG. 4

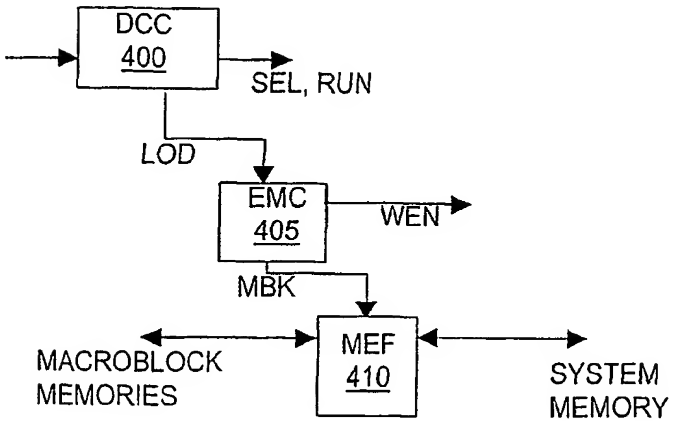

DCC
400
SEL, RUN
LOD
EMC
405
WEN
MBK
MEF
410
MACROBLOCK MEMORIES
SYSTEM MEMORY

FIG. 5

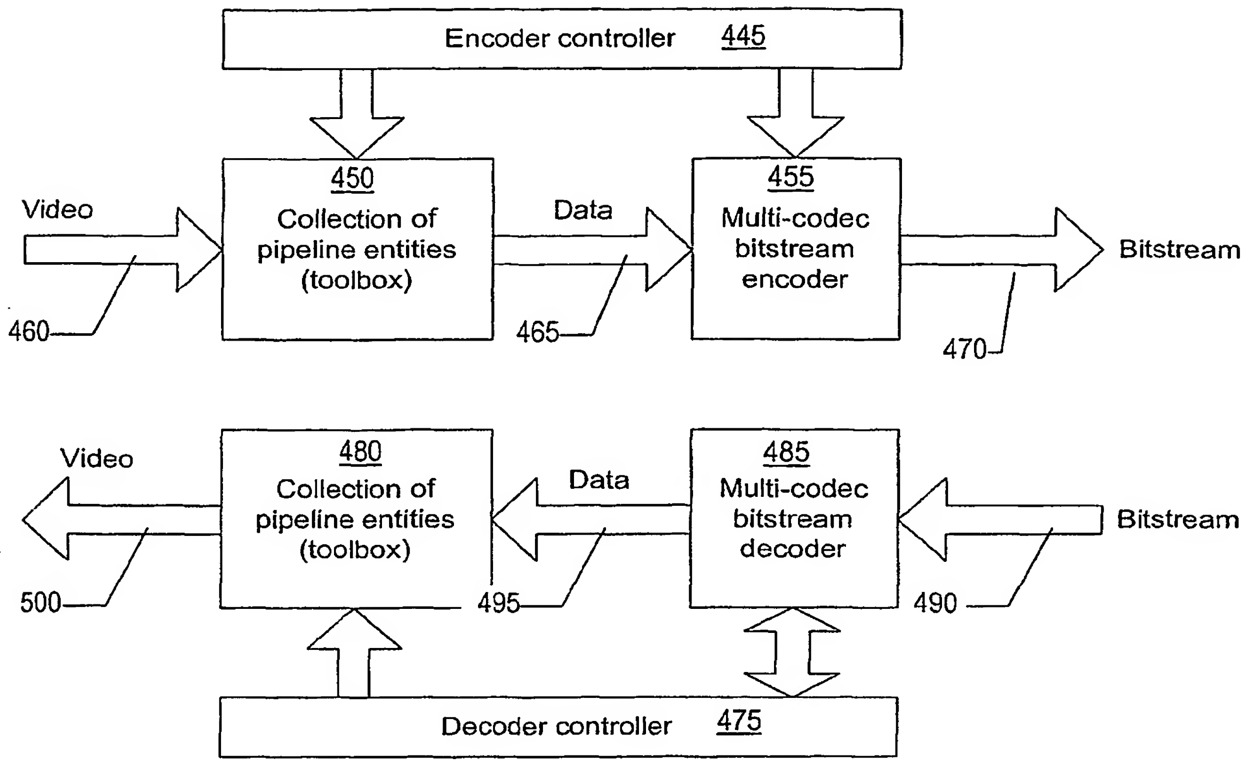

Encoder controller 445
450
Collection of pipeline entities (toolbox)
455
Multi-codec bitstream encoder
Video
Data
Bitstream
460
465
470
480
Collection of pipeline entities (toolbox)
485
Multi-codec bitstream decoder
Video
Data
Bitstream
500
495
490
Decoder controller 475

FIG. 6

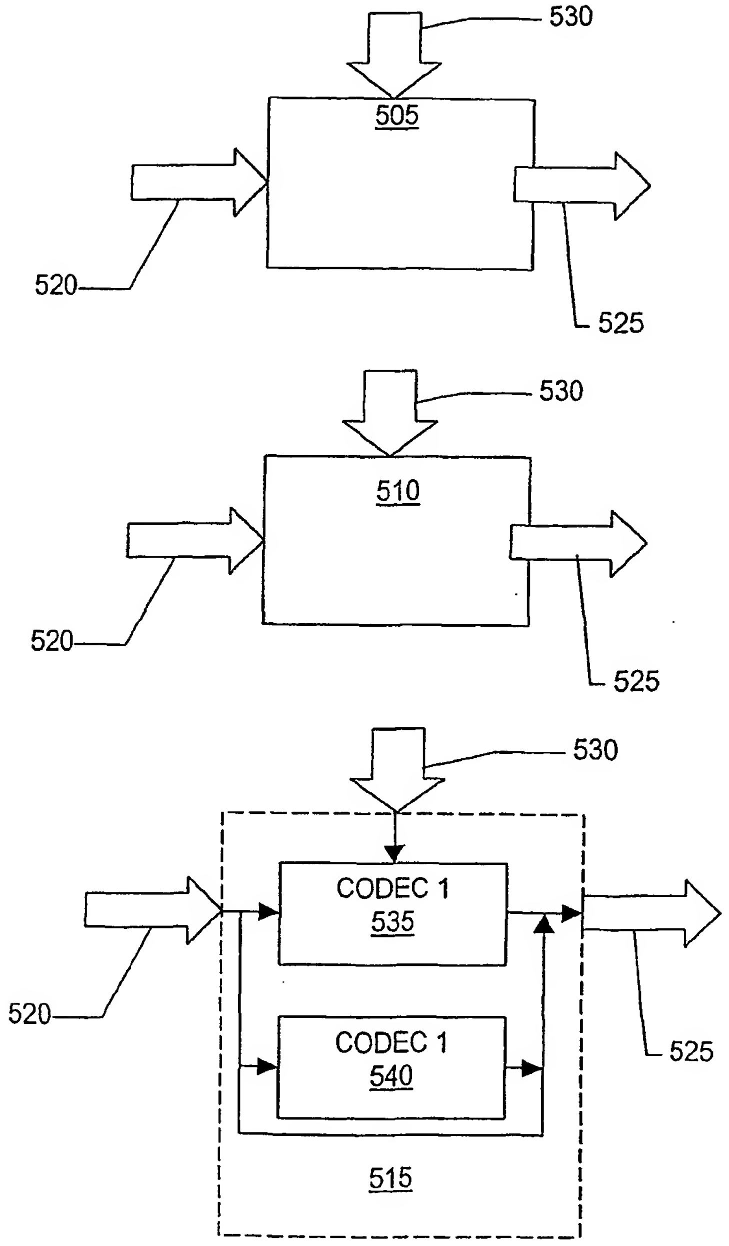

530
505
520
525
530
510
520
525
530
CODEC 1
535
CODEC 1
540
520
525
515

FIG. 7

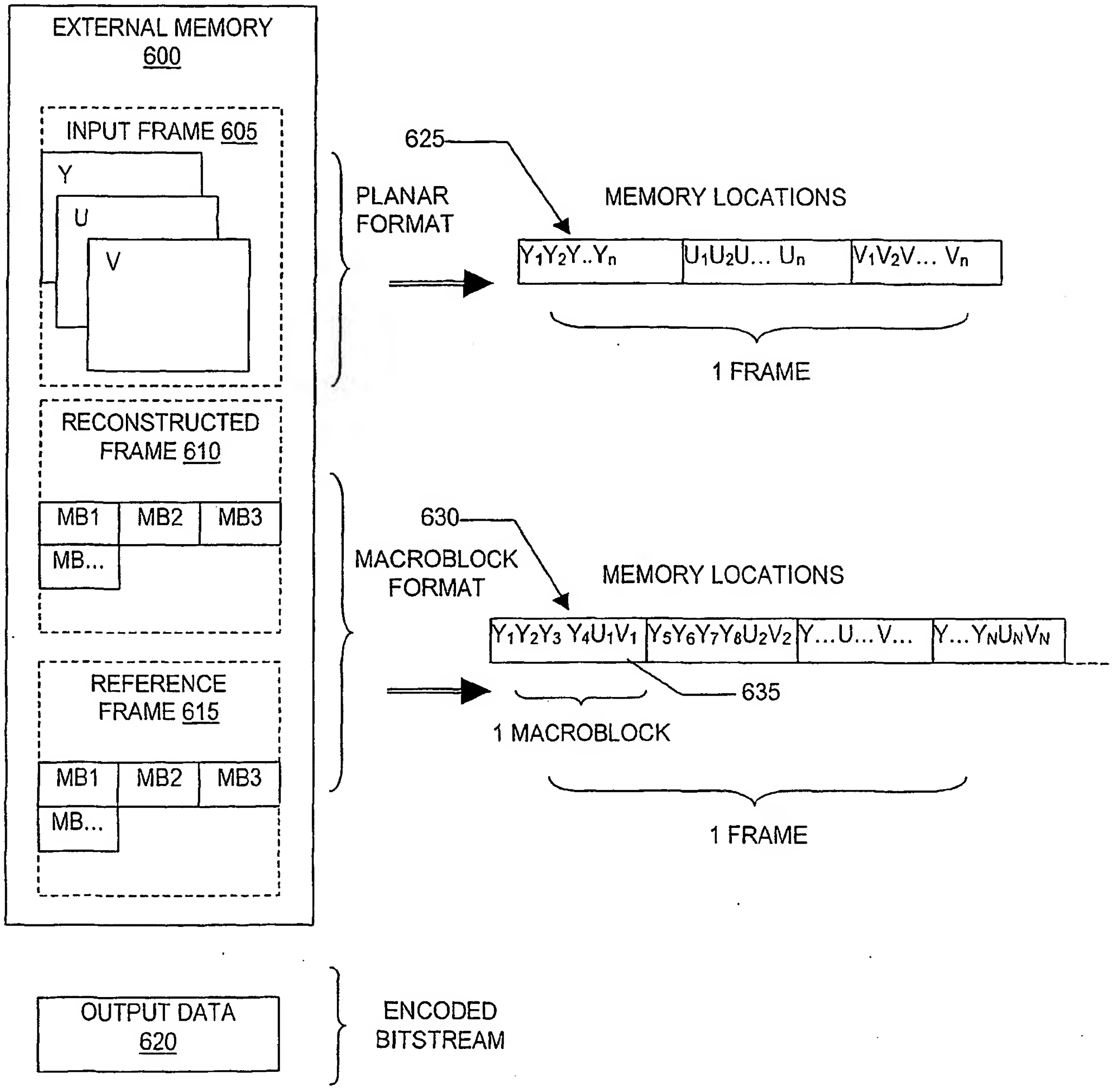

EXTERNAL MEMORY 600
INPUT FRAME 605
Y
U
V
RECONSTRUCTED FRAME 610
MB1
MB2
MB3
MB...
REFERENCE FRAME 615
MB1
MB2
MB3
MB...
OUTPUT DATA 620
625
PLANAR FORMAT
MEMORY LOCATIONS
Y1Y2Y..Yn
U1U2U... Un
V1V2V... Vn
1 FRAME
630
MACROBLOCK FORMAT
MEMORY LOCATIONS
Y1Y2Y3 Y4U1V1
Y5Y6Y7Y8U2V2
Y...U...V...
Y...YNUNVN
635
1 MACROBLOCK
1 FRAME
ENCODED BITSTREAM

FIG. 8

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
31 October 2002 (31.10.2002)

PCT

(10) International Publication Number
WO 02/087248 A3

(51) International Patent Classification⁷: H04N 7/50

(21) International Application Number: PCT/GB02/01796

(22) International Filing Date: 18 April 2002 (18.04.2002)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:

01303562.1	19 April 2001 (19.04.2001)	EP
60/305,846	18 July 2001 (18.07.2001)	US

(71) Applicant (*for all designated States except US*): INDIGOVISION LIMITED [GB/GB]; The Edimburgh Technopole, Bush Loan, Edimburgh EH26 0PJ (GB).

(72) Inventors; and
(75) Inventors/Applicants (*for US only*): SMART, Michael, Howard, William [GB/GB]; 1F3, 11 Templepark Crescent, Edimburgh EH11 1JF (GB). MACRAE, Donald, James [GB/GB]; Flat 11, 23 Crosscauseway, Edimburgh EH8 9JW (GB). DALZELL, Ryan [GB/GB]; 8/11 Starbank Road, Edimburgh EH15 3BW (GB). KEEPENCE, Barry [GB/GB]; 8 Merchiston Place, Edimburgh EH10 4NP (GB). BEATTIE, Robert, Scott [GB/GB]; Flat 2, 29 West High Street, Lauder, Tweeddale TD2 6TF (GB).

(74) Agent: FITZPATRICKS; 4 West Regent Street, Glasgow G2 1RS (GB).

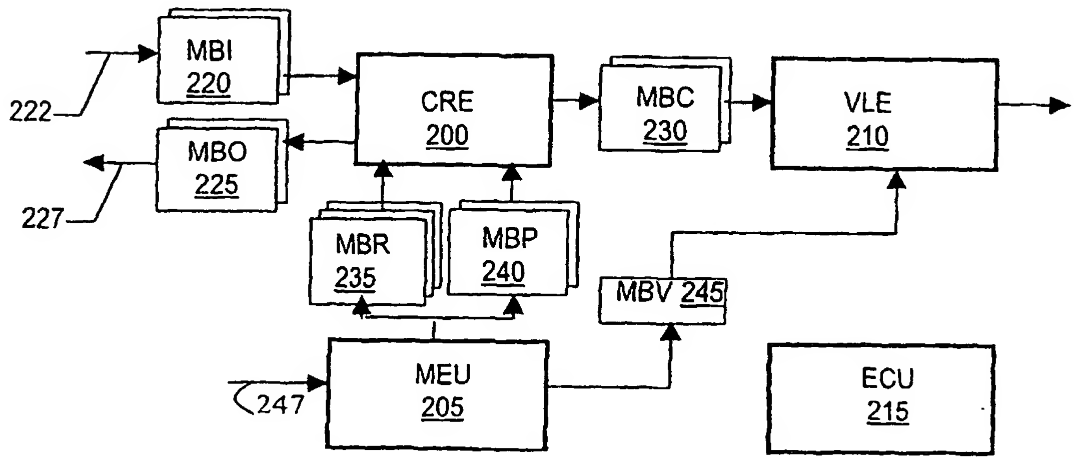
(81) Designated States (*national*): AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, OM, PH, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZM, ZW.

(84) Designated States (*regional*): ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *with international search report*

[Continued on next page]

(54) Title: APPARATUS AND METHOD FOR PROCESSING VIDEO DATA



(57) Abstract: The present invention relates to processing hardware suitable for enabling an integrated chip (5) solution for decoding and/or encoding video data in accordance with a variety of video encoding formats, comprising a pipeline controller (55, 215) and a plurality of processing stages (15-30, 200-210) arranged in a pipeline for processing successive blocks of data corresponding to successive blocks of an image being processed, at least one of said processing stages (15-30, 200-210) being arranged to process a block of data with reference to data from a previously processed image, data arriving at each processing stage at regular intervals, where it is modified and passed on to the next stage, and so on. The pipeline controller (55, 215) may be arranged to specify one of a number of different encoding protocols supported by the pipeline processor, at least one of said processing stages (15-30, 200-210) being arranged to configure itself differently for compatibility with the specified protocol.

WO 02/087248 A3

— *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments*

(88) Date of publication of the international search report:
19 December 2002

INTERNATIONAL SEARCH REPORT

Int. onal Application No
PCT/GB 02/01796

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 H04N7/50

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7 H04N

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)
EPO-Internal, INSPEC, COMPENDEX

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	OGURA E ET AL: "A 1.2-W SINGLE-CHIP MPEG2 MP ML VIDEO ENCODER LSI INCLUDING WIDE SEARCH RANGE (H:+/-288, V:+/-96) MOTION ESTIMATION AND 81-MOPS CONTROLLER" IEEE JOURNAL OF SOLID-STATE CIRCUITS, IEEE INC. NEW YORK, US, vol. 33, no. 11, November 1998 (1998-11), pages 1765-1771, XP000875470 ISSN: 0018-9200	1-7,9, 12-15, 20,22, 23,27,28
Y	abstract paragraph '000I! paragraph '00II!	24
A	figures 1,8,12	8,10,11, 16-19, 21,24-26

-/--

[X] Further documents are listed in the continuation of box C. [ ] Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier document but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.
"&" document member of the same patent family

Date of the actual completion of the international search 21 October 2002	Date of mailing of the international search report 30/10/2002
Name and mailing address of the ISA European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Authorized officer Montanari, M

INTERNATIONAL SEARCH REPORT

International Application No
PCT/GB 02/01796

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication,where appropriate, of the relevant passages	Relevant to claim No.
Y	SUBROTO BOSE ET AL: "A SINGLE CHIP MULTISTANDARD VIDEO CODEC" PROCEEDINGS OF THE CUSTOM INTEGRATED CIRCUITS CONFERENCE. SAN DIEGO, MAY 9 - 12, 1993, NEW YORK, IEEE, US, vol. CONF. 15, 9 May 1993 (1993-05-09), pages 110401-110404, XP000409686	24
A	the whole document	1-23, 25-28
X	CHEN G-L ET AL: "VIDEO ENCODER ARCHITECTURE FOR MPEG2 REAL TIME ENCODING" IEEE TRANSACTIONS ON CONSUMER ELECTRONICS, IEEE INC. NEW YORK, US, vol. 42, no. 3, 1 August 1996 (1996-08-01), pages 290-299, XP000638505 ISSN: 0098-3063 abstract paragraph ‘02.3!	1-7, 12-15, 22,23
A	figures 4,5	8-11, 16-21, 24-28
A	FERNANDEZ J M ET AL: "A HIGH-PERFORMANCE ARCHITECTURE WITH A MACROBLOCK-LEVEL-PIPELINE FOR MPEG-2 CODING" REAL-TIME IMAGING, ACADEMIC PRESS LIMITED, GB, vol. 2, no. 6, 1 December 1996 (1996-12-01), pages 331-340, XP000656194 ISSN: 1077-2014 abstract page 332, paragraph ARCHITECTURE figures 3,5,9	1-28
A	ARAKI T ET AL: "VIDEO DSP ARCHITECTURE FOR MPEG2 CODEC" PROCEEDINGS OF THE INTERNATIONAL CONFERENCE ON ACOUSTICS, SPEECH, ANDSIGNAL PROCESSING (ICASSP). SPEECH PROCESSING 2, AUDIO, UNDERWATER ACOUSTICS, VLSI AND NEURAL NETWORKS. ADELAIDE, APR. 19 - 22, 1994, PROCEEDINGS OF THE INTERNATIONAL CONFERENCE ON, vol. 2 CONF. 19, 19 April 1994 (1994-04-19), pages II-417-II-420, XP000528506 ISBN: 0-7803-1776-9 abstract paragraph ‘02.1! figure 2	1-28